

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 05-257902

(43)Date of publication of application : 08.10.1993

(51)Int.Cl.

G06F 15/16

G06F 9/46

(21)Application number : 04-051948

(71)Applicant : FUJITSU LTD

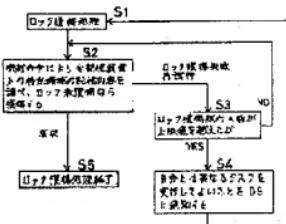
(22)Date of filing : 10.03.1992

(72)Inventor : KATO YOSHIO

(54) LOCK ACQUISITION PROCESSING SYSTEM IN PROCESSING PROGRAM MODE

(57)Abstract:

PURPOSE: To provide a lock acquisition processing system capable of acquiring lock by small overhead without reducing the efficiency of the whole system in respect to lock acquisition processing in a processing program mode for technological calculation or the like. CONSTITUTION: The stored contents of a specific area in a main storage device are checked by a machine instruction, and when lock is not acquired, the lock is acquired (S2). If lock acquisition is failed, lock acquisition trial frequency is counted (S3), and when the frequency exceeds a previously determined upper limit value, a message indicating the allowance of executing a task different from a self-task is informed to an OS (S4). Since the system is constituted so that said message is sent to the OS when the lock acquisition trial frequency exceeds the upper limit value, another program is not continuously looped for a long time until the lock is unlocked when a CPU is taken by the OS while holding the lock state, preventing the whole system from being dropped at its efficiency.



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's

(19)日本国特許庁 (JP)

(12) 公開特許公報 (A)

(11)特許出願公開番号

特開平5-257902

(43)公開日 平成5年(1993)10月8日

(51)Int.Cl. ⁶	識別記号	序内整理番号	F I	技術表示箇所
G 0 6 F 15/16	3 5 0 F	8840-5L		
9/46	3 4 0 F	8120-5B		

審査請求 未請求 請求項の数2(全13頁)

(21)出願番号 特願平4-51948
 (22)出願日 平成4年(1992)3月10日

(71)出願人 000005223
 富士通株式会社
 神奈川県川崎市中原区上小田中1015番地
 (72)発明者 加藤 喜郎
 神奈川県川崎市中原区上小田中1015番地
 富士通株式会社内
 (74)代理人 弁理士 京谷 四郎

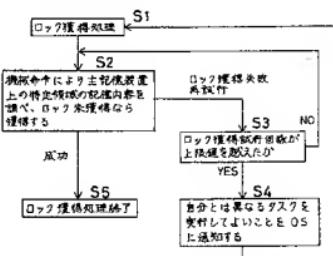
(54)【発明の名称】 構成プログラム・モードにおけるロック獲得処理方式

(57)【要約】

【目的】 科学技術計算等の処理プログラム・モードにおけるロック獲得処理において、システム全体の効率を低下させることなく、また、少ないオーバヘッドでロックを獲得することができるロック獲得処理方式を提供すること。

【構成】 機械命令により、主記憶装置上の特定領域の記憶内容を調べ、ロック未獲得なら、ロックを獲得する。ロック獲得が失敗した場合には、ロック獲得試行回数をカウントし、ロック獲得試行回数が予め定められた上限値を越えたとき、自分とは異なるタスクを実行してもよいことをOSに通知する。ロック獲得試行回数が上限値をこえた際、OSに通知するように構成したので、ロックを保持したままOSにCPUを奪われた場合、ロックが解放されるのを待つ他のプログラムが長時間ループし続けることがなく、システム全体の効率の低下を防ぐことができる。

本発明の原理フローチャート



【特許請求の範囲】

【請求項1】 主記憶装置上の特定領域の記憶内容を調べてその領域の記憶内容を書き換えることができる機械命令を組み合わせてロックを獲得する処理プログラム・モードにおけるロック獲得処理方式において、ロック獲得試行回数をカウントし、ロック獲得試行回数が予め定められた上限値を超えたとき、自分とは異なるタスクを実行してもよいことをオペレーティング・システムに通知することを特徴とする処理プログラム・モードにおけるロック獲得処理方式。

【請求項2】 ロックを獲得した状態で走行するときの平均経過時間をT、一回の再試行に要する時間をtとしたとき、ロック獲得試行回数の上限値をn=T/tとするように選定することを特徴とする請求項1の処理プログラム・モードにおけるロック獲得処理方式。

【発明の詳細な説明】

【0001】

【産業上の利用分野】 本発明は主記憶共用型マルチ・プロセッサ・システムにより科学技術計算を並列処理する際、ユーザ・プログラム及び実行時ライブラリでシステム資源の排他制御を行うためのロック獲得方式に関する。

【0002】

【従来の技術】 科学技術計算等の性能が要求される応用プログラムにおいては、実効性能を向上させるため、D・ループやサブルーチンを複数の演算装置により並列に実行する処理方式が用いられる。図8は上記処理を行う場合のシステム構成を示す図であり、同図において、1 0 1 は主記憶装置、1 0 2 ないし 1 0 5 は演算装置である。

【0003】 図8に示すシステム構成の計算機システムにおいては、複数の演算装置1 0 2 ないし 1 0 5 からアクセス可能な主記憶装置1 0 1 にプログラム及びデータをロードし、複数の演算装置1 0 2 ないし 1 0 5 が各々担当すべき部分のプログラムを実行する。図8(a)および(b)に示すシステムにおいては、理想的には、1台の演算装置により元のプログラムを実行したときの1/2(図8(a)の元)や1/4(図8(b)の場合)の処理時間でジョブの実行が終了することになる。

【0004】 しかしながら、実際には、実行効率はプログラムの構造(並列実行が可能なループがない場合等)と並列処理時のオーバヘッドにより大きな影響を受ける。このため、ユーザ・プログラムを並列実行可能な構造に書き直したり、あるいは並列処理制御のオーバヘッドを減らしたりするなど、ユーザ・プログラムのチューニングあるいはシステムのチューニングを行うことにより実行効率を向上させる必要があり、それによつて理想的な性能に近づけることが可能となる。

【0005】 上記システムのチューニングの1つの手法としては、システム資源の排他制御におけるオーバヘ

ドの低減がある。そこで、つぎに、本発明の前提となる並列処理におけるシステム資源の排他制御について説明する。図9は2つのサブルーチンによる並列処理を示す図である。例えば下記のようにメインルーチンSAMPLEよりサブルーチンSUB1, SUB2を呼び出し、サブルーチンSUB1, SUB2により並列処理を行う場合、図9に示すように、まず、図9の*1において並列処理の開始処理のための実行時ライブラリを呼び出す。

【0006】

PROGRAM SAMPLE

```
10    1000 PARCALL
          CALL SUB1(A, B, C)
          CALL SUB2(X, Y, Z)
1000 1000 END PARCALL
          PRINT*, A, X
          *****
```

このとき、サブルーチンSUB1, SUB2は実行時ライブラリによって、タスクにスケジュールされる。そして、サブルーチンSUB1, SUB2の実行が行われると、図9の*2に示すように、メインルーチンSAMPLEはサブルーチンSUB1, SUB2の実行が完了するまで待ち状態になる。

【0007】 ついで、サブルーチンSUB1, SUB2の実行が終わると、図9の*3に示すように、並列処理の終了処理が行われる。この処理が完了すると、メイン・ルーチンSAMPLEの処理が再開する。図10はサブルーチンSUB1, SUB2の並列実行開始時の処理手順を示す図である。同図において、1 1 1 は主記憶装置、1 1 1 a はキューリストラクチャ、1 1 1 b はサブルーチンSUB1の制御ブロック、1 1 1 c はサブルーチンSUB2の制御ブロック、1 1 2 は第1のタスク、1 1 2 a は第1のタスクの実行時ライブラリ、1 1 3 は第2のタスク、1 1 3 a は第2のタスクの実行時ライブラリである。

【0008】 プログラムが実行を開始する時、初期化処理の段階で、タスクを生成(ATTACH)し、タスクの初期化処理を行う。初期化完了直後は、実行すべきプログラム部分がないので、タスクは待ち状態になっている。サブルーチンSUB1とサブルーチンSUB2の並列処理は次の手順で並列実行が始まる。

(A) 主記憶装置1 1 1 上にあるサブルーチンSUB1に対応する制御ブロック1 1 1 b を初期化する。初期化処理の一部として、制御ブロックの中に、入口点アドレス、パラメータのアドレス、復帰点アドレス等の情報を格納する。

(B) サブルーチンSUB2の制御ブロック1 1 1 c も同様に初期化する。

(C) 初期化済の制御ブロック1 1 1 b 、1 1 1 c を主記憶装置1 1 1 上のキューリストラクチャに接続する。

(D) 待ち状態になっているタスクに対して、実行可能な手順がキューリストラクチャに存在することを通知する。

でロック獲得処理を行う場合には割込みをマスクしてロック獲得処理を行うことができる、上記のような問題は発生しないが、処理プログラム・モードでは割込みをマスクすることが出来ないので、システム効率の悪化に対処することができない。以上示したように、従来用いられていました排他制御機構には上記した種々の問題があり、科学技術計算等の処理プログラム・モードにおけるオーバーヘッドに適切に対処することができなかつた。

【 0 0 1 9 】

【 発明が解決しようとする課題】本発明は上記した従来技術の欠点を改善するためになされたものであって、科学技術計算等の処理プログラム・モードにおけるロック獲得処理において、システム全体の効率を低下させることなく、また、少ないオーバーヘッドでロックを獲得することができるロック獲得処理方式を提供することを目的とする。

【 0 0 2 0 】

【 課題を解決するための手段】図1は本発明の原理フローチャートである。本発明は、上記課題を解決するため、図1に示すように、主記憶装置上の特定領域の記憶内容を調べ、特定領域の記憶内容を書き換えることができる機械命令を組み合わせてロックを獲得する処理プログラム・モードにおけるロック獲得処理方式において、ロック獲得試行回数をカウントし、ロック獲得試行回数が予め定められた上限値を越えたとき、自分とは異なるタスクを実行してもよいことをOSに通知するように構成したものである。

また、上記構成に加え、獲得した状態で走行するときの平均経過時間を T 、一回の再試行に要する時間を τ としたとき、ロック獲得試行回数の上限値 n を $n = T / \tau$ となるように選定することができる。

【 0 0 2 1 】

【 作用】ロック獲得試行回数をカウントし、ロック獲得試行回数が上限値をこえると、自分とは異なるタスクを実行してもよいことをOSに通知するように構成したので、ロックを保持したままOSにCPUを奪われた場合、ロックが解放されるのを待って他のプログラムが長時間ループし続けることがなく、システム全体の効率を低下させることなく処理を進めることができる。

【 0 0 2 2 】また、上記上限値として、ロックを獲得した状態で走行するときの平均経過時間を T 、一回の再試行に要する時間を τ 、試行回数カウンタのカウント値の上限値を n とすると、 $T = \tau \times n$ 、すなわち、 $n = T / \tau$ となるように選定することにより、システム全体の効率の最適化を図ることができる。

【 0 0 2 3 】

【 実施例】図2は本発明の実施例を示すフローチャートである。本実施例は機械命令(TS, CS, CDS)を用いたロック獲得処理(前記した③)をベースにしたものであり、同図に示すように、ステップS10において

て、ロック獲得の試行回数カウンタを0に初期化したのも、ステップS11において、ロック・ワードを調べ、ロックが未獲得ならTS, CS, CDS命令を使ってロック獲得処理を行ふ。

【 0 0 2 4 】ステップS11において、ロックの獲得に失敗すると、ステップS12に行きロック獲得の試行回数カウンタを1増加する。ついで、ステップS13において、ロック獲得の試行回数カウンタのカウント値が上限値を越えたか否かを判別し、試行回数カウンタのカウント値が上限値を越えていない場合にはステップS11に戻り上記処理を繰り返す。

【 0 0 2 5 】また、ステップS13において、試行回数カウンタのカウント値が上限値を越えた場合には、ステップS14に行き、OSにCPUを返す。すなわち、自分とは異なるタスクが実行可能なら、そのタスクを実行してもよいことをOSに通知するためソフトウェア割込み命令であるSVC命令(スーパーバイザ・コール命令)を発行する。

【 0 0 2 6 】また、ステップS11において、ロック獲得に成功するとロック獲得処理を終了する。本実施例においては、ロック獲得試行回数をカウンタによりカウントし、試行回数が上限値をこえると、SVC命令を発行し自分と異なるタスクを実行してよいことをOSに通知するようにしたので、前記した③における問題点、すなわち、ロックを保持したままOSにCPUを奪われた場合、ロックが解放されるのを待って他のプログラムが長時間ループし続けることがなく、システム全体の効率を低下させることなく処理を進めることができる。

【 0 0 2 7 】なお、試行回数カウンタのカウント値が上限値を越えるとSVC割込みが発生しオーバーヘッドを生ずるが、上記上限値を適当な値に設定することにより、システムの効率の最適化を図ることができる。上記上限値としては、例えば、ロックを獲得した状態で走行するときの平均経過時間を T 、一回の再試行に要する時間を τ 、試行回数カウンタのカウント値の上限値を n とすると、 $T = \tau \times n$ 、すなわち、 $n = T / \tau$ となるように選定する。

【 0 0 2 8 】上記のように上限値を選定した場合、ロックを獲得できずに試行回数カウンタのカウント値が上限値 n を超過した場合には、ロックを獲得したタスクがロックを解放するまでにCPUを奪われた可能性が高いと判断することができる。したがって、試行回数が上限値 n を越えた場合に、ロックの解放を待っているタスクがいったんCPUをOSに返すことにより、そのタスクが長時間ループし続けることはなく、システム全体の効率の低下を防ぐことができる。

【 0 0 2 9 】また、一方、ロックを獲得してから解放するまで連続して処理が行われた場合には、その走行時間は上記平均経過時間 T より少ない場合が多いから、試行回数カウンタのカウント値が上限値 n を越える前に、待

ち状態は解消されることが期待できる。なお、上記、ロック獲得処理をマクロ化しておくことにより、科学技術計算等の並列処理のための実行時ライブラリの中でユーザーマクロとして用いることができる。

【 0 0 3 0 】以上のように、本実施例の方式はロックを持ったまま走行する区間が短い場合に適用することにより、オーバーヘッドを少なくすることができます。図3および図4は上記実施例を図9および図10に示した2つのサブルーチンによる並列処理に適用した場合のフローチャートであり、図3および図4において、ステップR10およびステップT2は図2の実施例に示したロック獲得処理に対応する。

【 0 0 3 1 】つぎに、図3、図4および図10により、上記実施例の適用例について説明する。サブルーチンSUB1とサブルーチンSUB2の並列処理は、前記したように、次の手順で並列実行が始まる。実行中のタスクにおいては、図3に示すように、ステップR1において、主記憶装置111（図10）上にあるサブルーチンSUB1に対応する制御ブロック111b（図10）を初期化する。初期化処理の一環として、制御ブロックの中に入口点アドレス、パラメタのアドレス、復帰点アドレス等の情報を格納する。

【 0 0 3 2 】ステップR2において、サブルーチンSUB2の制御ブロック111c（図10）も同様に初期化する。ステップR3において、キューのアクセス権を得るためにロックを獲得する。この際、ステップR4に行き、図2の実施例に示した処理を実行する。すなわち、ロック獲得処理が不成功であった場合の試行回数をカウントし、その上限値以内にロックが獲得できない場合には、ソフトウェア割込み命令であるSVC命令を発行し、自分とは異なるタスクが実行可能ならそのタスクを実行してみることをOSに通知する。そして、自分とは異なるタスクの実行が終了すると、再び、ロック獲得処理を行い、ロックを獲得する。

【 0 0 3 3 】また、試行回数の上限値以内にロック獲得に成功した場合には、ステップR5に行き、初期化済の＊

*制御ブロック111b、111c（図10）を主記憶装置111上のキュー111a（図10）に接続する。ついで、ステップR6において、キューのアクセス権を放棄するためにロック解放処理を行い、ステップR7において、待ち状態になっているタスクに対して、実行可能な手続きがキュー111a（図10）に存在することを通知する。

【 0 0 3 4 】実行可能な手続きの存在を通知されたタスクにおいては、図4に示すように、ステップT1において、キューのアクセス権を得るためにロック獲得処理を行う。この際、前記したと同様に、ステップT2に行き、図2の実施例に示した処理を実行する。すなわち、ロック獲得処理が不成功であった場合の試行回数をカウントし、その上限値以内にロックが獲得できない場合には、ソフトウェア割込み命令であるSVC命令を発行し、自分とは異なるタスクが実行可能ならそのタスクを実行してみることをOSに通知する。

【 0 0 3 5 】また、試行回数の上限値以内にロック獲得に成功した場合には、ステップT3に行き、キュー111a（図10）を探索して制御ブロックを取得する。ついで、ステップT4において、キューのアクセス権を放棄するためにロック解放処理を行い、ステップT5において、制御ブロックの中の情報を取り出して、実行すべき手続きを呼び出す。

【 0 0 3 6 】図5、図6および図7は図2に示したロック獲得処理を下記の並列処理プログラムの実行時ライブラリにおけるキュー管理に適用した場合のフローチャートである。図5、図6および図7において、ステップP5、ステップQ5およびステップU2は図2の実施例に示したロック獲得処理に対応する。上記キュー管理は、FORTRANソース・プログラム中に、下記のように、並列実行中の他手続きと同期を取るための命令（この命令を「p-barrier」とする）を記述したサブルーチン（p-barrier）を設け、そのプログラムを実行した場合に必要となる。

【 0 0 3 7 】

PROGRAM MAIN	SUBROUTINE SUB1(A,B,C)	SUBROUTINE SUB2(X,Y,Z)
...
! OCL PARCALL	DO 1,I=...	CALL p-barrier
CALL SUB1(A,B,C)	1 A(I)=...	...
CALL SUB2(X,Y,Z)	...	DO 1,I=...
! OCL END PARCALL	CALL p-barrier	1 X(I)=...
PRINT*,A,X
...	END	END
END		

上記プログラムにおける「CALL p-barrier」の行は、サブルーチンSUB1とSUB2の並列実行中に待ち合わせを行るために設けられたものであり、「CALL p-barrier」による待ち合わせを実現するため、待ち状態の手続きの制御ブロックを接続しておくためのキューを使用する。この

キューは呼び出し元（上記プログラムの例においては、PROGRAM MAIN）の制御ブロックの中にある。

【 0 0 3 8 】そして、制御ブロックの中には、初期化のときに、並列呼び出しの呼び出し元の手続きの制御ブロックのアドレスを格納してあるので、サブルーチンSUB1

及びSUE2からp-barrierのキューを参照することができる。また、呼び出し元の制御ブロックには、並列呼び出しを行った手続きの数(この例では2)とp-barrierを実行して待ち状態になっている手続きの数(初期値は0)を格納するフィールドがある。

【0039】つぎに図5、図6および図7により、図2に示したロック獲得処理の第2の適用例について説明する。サブルーチンSUE1とSUE2が並列動作中のとき、サブルーチンSUE2の中で、サブルーチンSUE1よりも先にp-barrierが呼び出された場合を想定すると、図5に示すように、サブルーチンSUE2のp-barrierにおいては、ステップP1において、呼び出し元(MAIN)の制御ブロックを参照する。

【0040】そして、p-barrierで待ち状態になっている手続きの数が0であることから、p-barrierを実行したのはサブルーチンSUE2が最初であることが判明し、ステップP2において、p-barrier待ち状態の手続きの数を1増加する。ステップP3において、SUE2の制御ブロックに再開アドレスを格納して、SUE2を待ち状態にする。

【0041】ついで、ステップP4において、ロックを獲得する。その際、前記したのと同様に、ステップP5に行き、図2の実施例に示した処理を実行する。すなわち、ロック獲得処理が不成功であった場合の試行回数をカウントし、その上限値以内にロックが獲得できない場合には、ソフトウェア割込み命令であるSVC命令を発行し、自分とは異なるタスクが実行可能ならそのタスクを実行してよいことをOSに通知する。

【0042】また、試行回数の上限値以内にロック獲得に成功した場合には、ステップP6に行き、p-barrierの待ち用のキューにSUE2の制御ブロックを接続し、ステップP7において、ロックを解放する。つぎに、サブルーチンSUE1のp-barrierにおいては、ステップQ1において、呼び出し元(MAIN)の制御ブロックを参照する。そして、p-barrierで待ち状態になっている手続きの数が1であることから、p-barrierを実行したのはサブルーチンSUE1が最後で、既に待ち状態の制御ブロックが1個p-barrierの待ち用のキューにつながっていることが判明し、ステップQ2において、p-barrier待ち状態の手続きの数を0にリセットする。

【0043】ステップQ3において、p-barrier待ち用のキューからサブルーチンSUE2の制御ブロックをはずす。すなわち、並列処理中にp-barrier待ち用のキューをアクセスするにはサブルーチンSUE1、SUE2の2者しかないので、サブルーチンSUE1は自分が最後であることをこの時点できつていているから、排他制御をする必要がない。

【0044】ついで、ステップQ4において、ロックを獲得する。その際、前記したのと同様に、ステップQ5に行き、図2の実施例に示した処理を実行する。すなわち、

ロック獲得処理が不成功であった場合の試行回数をカウントし、その上限値以内にロックが獲得できない場合には、ソフトウェア割込み命令であるSVC命令を発行し、自分とは異なるタスクが実行可能ならそのタスクを実行してもよいことをOSに通知する。そして、自分とは異なるタスクの実行が終了すると、再び、ロック獲得処理を行い、ロックを獲得する。

【0045】また、試行回数の上限値以内にロック獲得に成功した場合には、ステップQ6に行き、並列実行開始時に接続したキューに接続し、ステップQ7において、ロックを解放する。ステップQ8において、待ち状態になっているタスクに対して実行可能な手続きがキューに存在することを通知し、ステップQ9において、p-barrierの呼び出し元に復帰してサブルーチンSUE1の実行を続ける。

【0046】ステップQ8において、実行可能な手続きがキューに存在することを通知されたタスク(サブルーチンSUE2のp-barrier)においては、図7のフローチャートに示すように、ステップU1において、ロックを獲得する。その際、前記したのと同様に、ステップU2に行き、図2の実施例に示した処理を実行し、ロックを獲得する。

【0047】ステップU3において、キューを探索して、制御ブロックを取得(キューからはずす)し、ステップU4において、ロックを解放する。ついで、ステップU5において、制御ブロックから再開アドレスを取り出して制御を渡し、サブルーチンSUE2を再開する。ステップU6において、p-barrierの呼び出し元に復帰して、サブルーチンSUE2の実行を続ける。

【0048】以上、図2の実施例の適用例として、2つのサブルーチンによる並列処理、および、並列処理プログラムの実行時ライブラリにおけるキュー管理に適用した場合を示したが、本発明の適用対象は上記適用例に限定されるものではなく、本発明は、上記適用対象以外の処理プログラム、モードにおけるロック獲得処理に適用することができる。

【0049】

【発明の効果】以上説明したことから明らかのように、本発明は、機械命令(TS、CS、CDS)を用いてロック獲得処理を行って、ロック獲得試行回数をカウントによりカウントし、試行回数が上限値をこえると、SVC命令を発行し、自分と異なるタスクを実行してよいことをOSに通知するようにしたので、処理プログラム、モードにおけるロック獲得処理において、システム全体の効率を低下させることなく、また、少ないオペレーティングシステムにおいてもロックを獲得することができる。

【図面の簡単な説明】

【図1】本発明の原理フローチャートである。

【図2】本発明の実施例のフローチャートである。

【図3】本発明の実施例の第1の適用例のフローチャートである。

トである。

【図4】本発明の実施例の第1の適用例のフローチャート(続き)である。

【図5】本発明の実施例の第2の適用例のフローチャートである。

【図6】本発明の実施例の第2の適用例のフローチャート(続き)である。

【図7】本発明の実施例の第2の適用例のフローチャート(続き)である。

【図8】主記憶共用型マルチ・プロセッサ・システムの構成を示す図である。

【図9】2つのサブルーチンによる並列処理を示す図である。

【図10】2つのサブルーチンによる並列処理実行開始時の処理手順を示す図である。

【図11】従来のロック獲得処理を示すフローチャートである。

【符号の説明】

1 1 1 主記憶装置

1 1 1 a キュー

1 1 1 b サブルーチンSUB1の制御ブロック

1 1 1 c サブルーチンSUB2の制御ブロック

1 1 2 第1のタスク

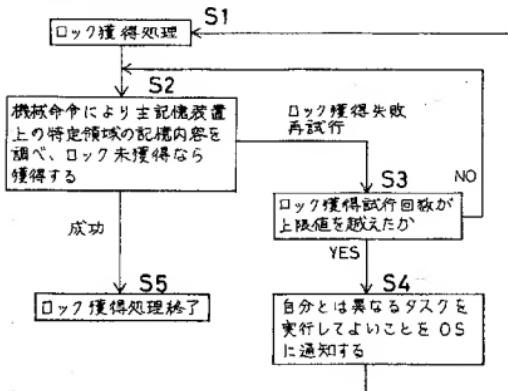
1 1 2 a 第1のタスクの実行時ライブラリ

1 1 3 第2のタスク

1 1 3 a 第2のタスクの実行時ライブラリ

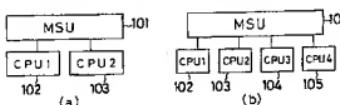
【図1】

本発明の原理フローチャート



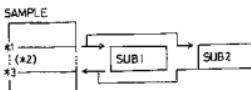
【図8】

主記憶共用型マルチ・プロセッサ・システムの構成を示す図



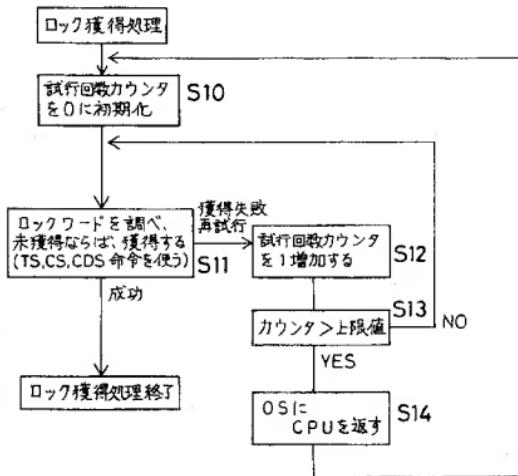
【図9】

2つのサブルーチンによる並列処理を示す図



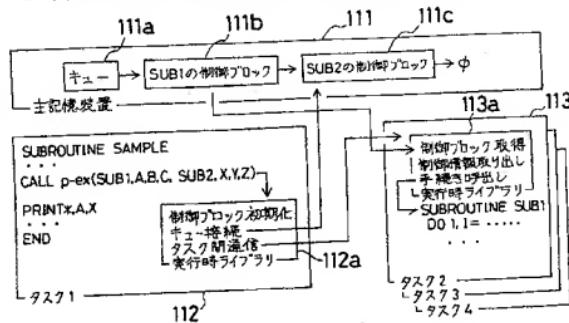
〔図2〕

本発明の実施例のフローチャート



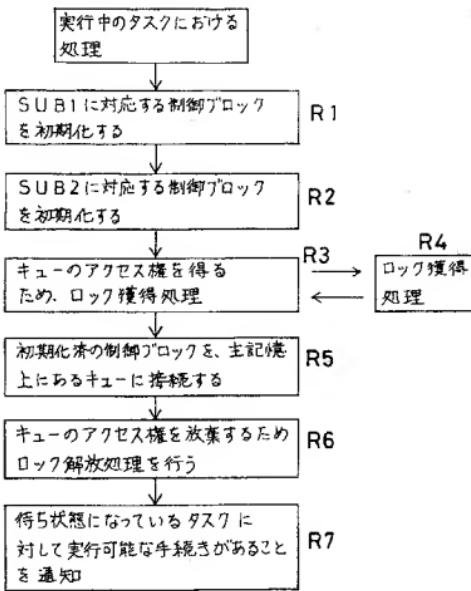
[10]

2つのサブループに於ける並列処理実行開始時の処理手順を示す図



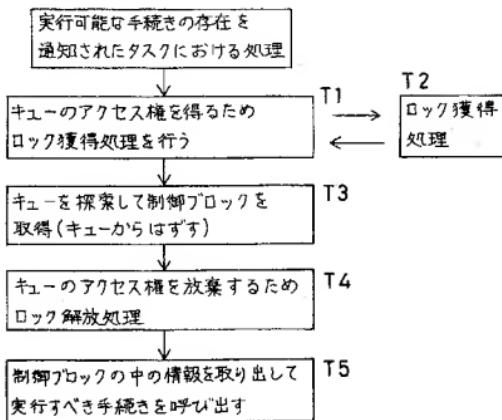
【 図3 】

本発明の実施例の第1の適用例のフローチャート



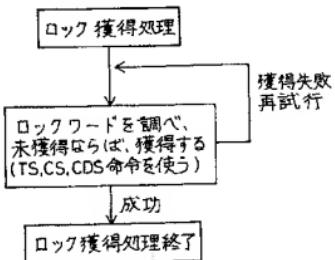
【図4】

本発明の実施例の第1の適用例のフロー・チャート(続き)



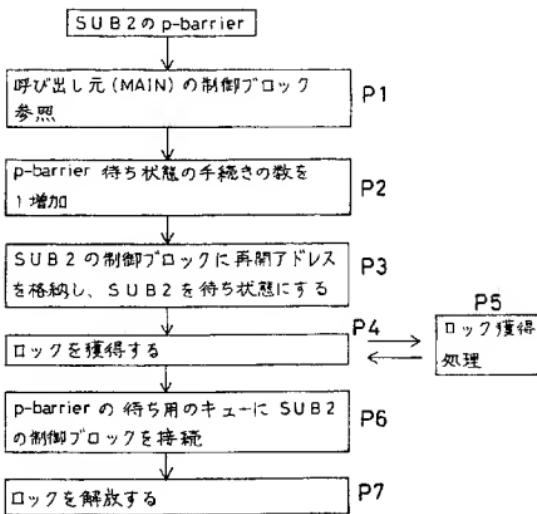
【図11】

従来のロック獲得処理を示すフロー・チャート



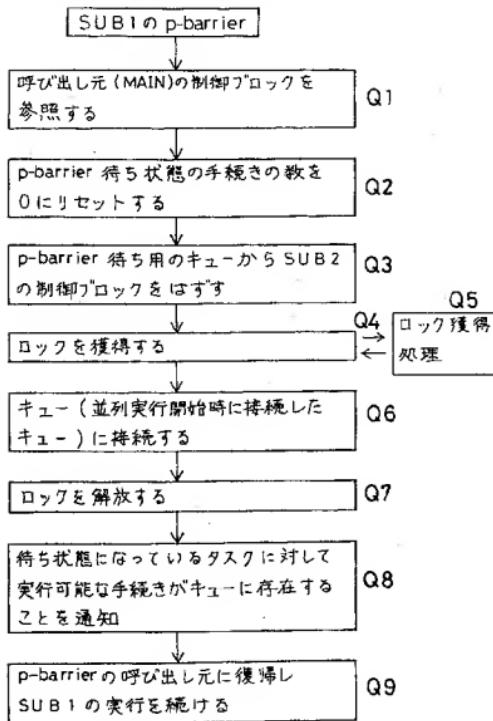
【 図5 】

本発明の実施例の第2の適用例を示すフローチャート



【 図6 】

本発明の実施例の第2の適用例のフローチャート(続き)



【図7】

本発明の実施例の第2の適用例のフローチャート(続き)

